

INSIDE DOS

Tips & techniques for MS-DOS & PC-DOS

Loading an application and a file in a single step using DOS 5's Shell

VERSION
5.0

If you've spent much time working with DOS 5's Shell, you know that it provides a congenial interface and some handy shortcuts, especially for DOS newcomers. Whether you issue Shell commands from the keyboard or with a mouse, the Shell can simplify common tasks.

One such feature the Shell offers is the ability to "associate" a filename extension with an application program. When you associate a program with an extension, you can load that program and any file bearing that extension in a single step. For example, if you associate the DOC filename extension with Microsoft Word, you can load Word and open any DOC file in a single step.

In this article, we'll show you how to associate a filename extension with an application program. We'll then show you how to launch the program and load a file using both the keyboard and the mouse. As you'll see, associating a file with a program is a snap, and once you do it, loading both is just a double-click away.

The Associate... command

The command you use to associate a filename extension with an application program is, appropriately enough, the Associate... command on the File menu. You can link a program and an extension in one of two ways:

1. You can select a file that contains the extension you want to associate, then tell DOS which program to associate it with.
2. You can select the program's executable file, then tell DOS which filename extension to associate it with.

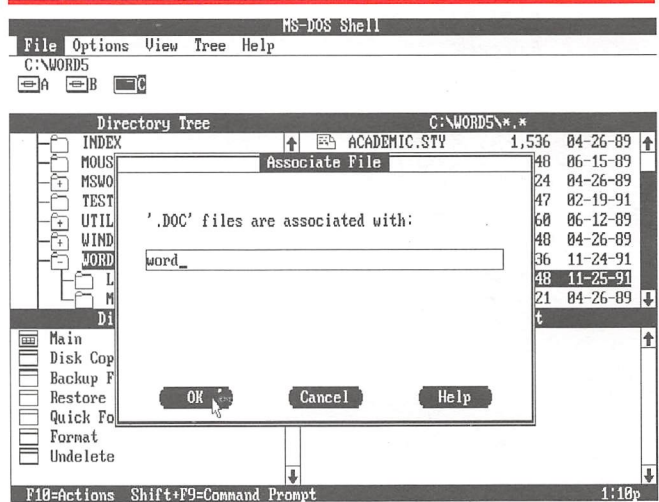
Let's take a look at both methods.

Associating the program with the extension

To associate a program and an extension by selecting the file, first highlight any filename containing the extension you want to use and issue the Associate... command on the File menu. When DOS presents the Associate File dialog box, type the name of the program's executable file with which you want to associate the extension (including the pathname if you want to use this technique across dir-

ectories). Next, press [Enter] or click OK, and DOS will associate the program with the extension.

Figure A



Using the Associate... command you can associate a filename extension with an application program, enabling you to load an application and a file in a single step.

For example, let's associate the DOC filename extension with Microsoft Word. First, select the \WORD5 directory in the Directory Tree window. When DOS displays the \WORD5 directory's files in the File List window, select any file containing the DOC extension and issue the Associate... command on the File menu. In the Associate File dialog box, type *word* (the command you use to load Microsoft Word),

Continued on page 11

IN THIS ISSUE

- Loading an application and a file in a single step using DOS 5's Shell 1
- Deleting directories and subdirectories 2
- Taking advantage of DOS' wildcards 4
- Creating a safer DEL command 6
- Van Wolverton: Treat your hard disk with respect—and care 7
- Double your hard drive space with Stacker 2.0 10

INSIDE DOS

Inside DOS (ISSN 1049-5320) is published monthly by The Cobb Group.

Prices	Domestic	\$59/yr. (\$7.00 each)
	Outside the US	\$79/yr. (\$8.50 each)
Address	The Cobb Group	
	9420 Bunsen Parkway, Suite 300 Louisville, KY 40220	
Phone	Toll-free	(800) 223-8720
	Local	(502) 491-1900
	FAX	(502) 491-8050
Staff	Editor-in-Chief	Jim Welp
	Consulting Editors	Mark W. Crane
		Tim Landgrave
	Contributing Editor	Van Wolverton
	Editing	Clyde Zellers
		Polly Blakemore
	Production	Debbie Lane
	Design	Karl Feige
	Publications Manager	Elayne Noltemeyer
	Publisher	Douglas Cobb

Address correspondence and special requests to The Editor, *Inside DOS*, at 9420 Bunsen Parkway, Suite 300, Louisville, KY 40220. Address subscriptions, fulfillment questions, and requests for bulk orders to Customer Relations, at the same address.

Postmaster: Send address changes to *Inside DOS*, P.O. Box 35160, Louisville, KY 40232. Second class postage is paid in Louisville, KY.

Copyright © 1992, The Cobb Group. All rights reserved. *Inside DOS* is an independently produced publication of The Cobb Group. No part of this journal may be reproduced in any form (except in brief quotations used in critical articles and reviews) without prior consent of The Cobb Group. Readers who wish to share information should write to The Editor, *Inside DOS*, 9420 Bunsen Parkway, Suite 300, Louisville, KY 40220. The Cobb Group reserves the right, with respect to submissions, to revise, republish, and authorize its readers to use the information submitted for both personal and commercial use.

The Cobb Group, its logo, and the Satisfaction Guaranteed statement and seal are registered trademarks of The Cobb Group. *Inside DOS* is a trademark of The Cobb Group. Microsoft is a registered trademark of Microsoft Corporation. IBM is a registered trademark of International Business Machines, Corporation.

Conventions

To avoid confusion, we would like to explain a few of the conventions used in *Inside DOS*.

When we instruct you to type something, those characters usually appear on a separate line along with the DOS prompt. The characters you type will appear in red, while the characters DOS displays will appear in black.

Occasionally, we won't display the command you type on a separate line. In these cases, we'll display the characters you type in italics. For example, we might say, "Issue the command *dir *.txt* at the DOS prompt." Although DOS is not case-sensitive, we'll always display the characters you type in lowercase.

When we refer to a general DOS command (not the command you actually type at the DOS prompt), we'll display that command name in all caps. For example, we might say, "You can use either the COPY or XCOPY command to transfer files from one disk to another."

Many commands accept parameters that specify a particular file, disk drive, or other option. When we show you the form of such a command, its parameters will appear in italics. For example, the form of the COPY command is

copy file1 file2

where *file1* and *file2* represent the names of the source file and the target file, respectively.

The names of keys, such as [Shift], [Ctrl], and [F1], appear in brackets. When two keys must be pressed simultaneously, those key names appear side by side, as in [Ctrl][Break] or [Ctrl]z.

LETTERS

Deleting directories and subdirectories

VERSIONS
2.1-5.0

I'm having trouble deleting a directory from my hard drive. I deleted all the files in the directory using the command

```
C:\TEST>del *.*
```

Then, I returned to the root directory using the command

```
C:\TEST>cd\
```

and issued the command

```
C:\>del test
```

DOS responded with the message

```
All files in directory will be deleted!  
Are you sure (Y/N)?
```

When I typed Y, DOS responded

```
File not found.
```

instead of deleting the directory. How do you delete a directory in DOS?

Russ Henderson
Norton, Ohio

The problem is that DOS' DEL command applies only to files. DOS provides a couple of other commands for handling directories. To create a directory, you need to use the command MD (or MKDIR), which stands for *Make Directory*. When you want to create a directory, just type the command *md* followed by the name you want to assign to the new directory.

The command you use to delete a directory in DOS is the RD (or RMDIR) command, which stands for *Remove Directory*. To remove an empty directory, you simply type at the directory's parent directory the command *rd* followed by the name of the directory you want to delete. For example, once you've deleted all the files in the \TEST directory, you can delete the directory itself by typing the command

```
C:\>rd test
```

and pressing [Enter]. However, before you can remove a directory, you must follow these rules:

- You can't remove a directory that contains hidden or system files.
- You can't remove a directory that contains subdirectories.
- You can't remove the current directory.

Let's take a look at how you can avoid these pitfalls and provide a couple of shortcuts for handling directories.

Deleting hidden or system files

You might have trouble deleting a directory because it contains hidden or system files. As you may know, DOS doesn't display hidden or system files—those with their hidden or system attributes turned on—in directory listings. Therefore, your directory might contain hidden or system files even though you've successfully carried out the `del *.*` command. For example, when you issue the command

```
C:\TEST>del *.*
```

and press *Y* in response to DOS' warning prompt, DOS won't delete any system or hidden files in the \TEST directory. In fact, if you confirm the delete operation with the `DIR` command

```
C:\TEST>dir
```

DOS' response

```
.          <DIR>  11-21-91      1:43p
..         <DIR>  11-21-91      1:43p
          2 file(s)         0 bytes
          73724872 bytes free
```

indicates that the global deletion was successful. What's worse, if you then return to the root directory and try to delete the \TEST directory by issuing the command

```
C:\>rd test
```

DOS responds with the message

```
Invalid path, not directory,
or directory not empty
```

which might not clue you in to the problem. To be able to delete the directory in DOS 5, you need to use the `ATTRIB` command to check for hidden or system files. To check for the presence of hidden or system files, type

```
C:\TEST>attrib
```

When you do, DOS will list all the files in the directory, preceded by any of four one-letter codes. If DOS lists any files preceded by an *H* or an *S*, that means that file's hidden or system attribute is turned on. In order to delete the file, you must turn off the hidden or system attribute.

For example, if the `ATTRIB` command turns up the hidden file

```
C:\TEST>attrib
A H      C:\TEST\SECRET.DOC
```

you must turn off that file's hidden attribute ("unhide" the file) before you can delete it, and therefore, be able to

delete the \TEST directory. Using DOS 5, you can turn off `SECRET.DOC`'s hidden attribute by typing

```
C:\TEST>attrib -h secret.doc
```

Once you do, you can delete the file, enabling you to use the `RD` command to delete the \TEST directory. Similarly, if your directory contains a system file, you can turn off that file's system attribute using `ATTRIB`'s *-S* parameter.

If you're using an earlier version of DOS, you can't turn off a file's hidden or system attribute with the `ATTRIB` command. There are some third-party utilities available that allow you to delete hidden or system files. Also, many hidden or system files that end up on your hard drive as a result of an application installation can be "uninstalled" by the application itself.

Removing subdirectories

As you might have guessed, the technique you use to delete subdirectories is the same as the one you use to delete directories: the `RD` command.

For example, if your \TEST directory contains a subdirectory called \DATA, you need to remove the \DATA subdirectory with the command

```
C:\TEST>rd data
```

before you can delete the \TEST directory. As you might expect, the same rules for deleting directories apply for deleting subdirectories: The directory must be empty of all files, including hidden and system files, and it must be empty of subdirectories.

Changing directories first

The final rule to keep in mind when deleting a directory is to make sure you're not trying to delete the current directory. For example, if you try to delete the \TEST directory while it's the current directory by issuing the command

```
C:\TEST>rd c:\test
```

DOS will respond with the error message

Attempt to remove current directory - C:\TEST

Before you issue the `RD` command, you should change directories. However, you don't need to be in the parent directory to delete a directory. You can delete a directory from another directory as long as you specify the full pathname. For example, you could delete the \TEST directory from the \BATCH directory with the command

```
C:\BATCH>rd c:\test
```

A note

By the way, if you want to delete a directory, you don't need to change to that directory and issue the command

`del *.*`, then return to the parent directory and issue the `RD` command. DOS allows you to delete all the files in a directory from the parent directory by issuing the `DEL` command in the form

```
del directory
```

where *directory* is the name of the directory that contains the files you want to delete. For example, to delete all the files in the `\TEST` directory, issue at the parent directory the command

```
C:\>del test
```

and press [Enter]. This command is the same as issuing the `del *.*` command from within the `\TEST` subdirectory. Because the command calls for deleting all the directory's files, DOS will display the warning prompt

```
All files in directory will be deleted!
Are you sure (Y/N)?
```

When you press *Y*, and then [Enter], DOS will delete all the files in the directory. You can then remove the directory without ever having had to change to it. ■

DOS FUNDAMENTALS

VERSIONS
2.1-5.0

Taking advantage of DOS' wildcards

You'll often want to use a DOS command to operate on more than one file at a time. For example, you might want to copy several files from one disk to another, or to erase a large group of files from a directory.

Fortunately, most DOS commands let you use wildcard characters to handle several files at once. This means that when you want to do the same thing to several files—such as copy or erase them—you don't need to enter a separate command for each file. Instead, you can use wildcards to specify a group of files that have similar names or extensions.

DOS provides two wildcards: the asterisk (*) and the question mark (?). Let's look at each one in detail.

The asterisk wildcard

The asterisk is DOS' most flexible (and by far its most often used) wildcard. You can use the asterisk to represent anything in a file's root name—from a single character to eight characters. Similarly, you can use the asterisk to represent anything in a file's extension from a single character to all three characters.

Let's look at a few examples of how you can specify files with the asterisk wildcard. For all of our examples, we'll use DOS' `DIR` command, which takes the form

```
dir filespec
```

where the optional argument *filespec* is a group of characters specifying the filenames you want to include in the listing. Of course, the result of the `DIR` command is a list of all the files in the current directory whose names meet the criteria specified by *filespec*. As you probably know, DOS will list every file in the current directory if you don't supply a *filespec* argument.

For each of the examples, assume the current directory contains only the files shown in Figure A.

Figure A

BUDGET91	TXT	3919	12-17-90	12:23p
DEPOSITS	TXT	3519	3-18-91	1:00p
REPORT	JAN	6434	2-13-91	10:04a
REPORT	FEB	6544	3-18-91	12:40p
REPORT	MAR	6134	4-11-91	3:30p
REPORT91	TXT	8485	1-18-91	11:20a
REVENUE	TXT	7357	3-18-91	12:00p

You can use wildcards to operate on groups of files that have similar names and extensions.

Examples

To list all the files in the current directory that have a `TXT` filename extension, append the filespec `*.txt` to the `DIR` command, like this:

```
C:\DATA>dir *.txt
```

and DOS will respond

```
Volume in drive C has no label
Directory of C:\DATA
```

BUDGET91	TXT	3919	12-17-90	12:23p
DEPOSITS	TXT	3519	3-18-91	1:00p
REPORT91	TXT	8485	1-18-91	11:20a
REVENUE	TXT	7357	3-18-91	12:00p

4 File(s) 10629120 bytes free

```
C:\DATA>_
```

Similarly, to list the files having the root name `REPORT`, use the filespec `report.*`:

```
C:\DATA>dir report.*
```

and DOS will respond

Volume in drive C has no label
Directory of C:\DATA

REPORT	JAN	6434	2-13-91	10:04a
REPORT	FEB	6544	3-18-91	12:40p
REPORT	MAR	6134	4-11-91	3:30p

3 File(s) 10629120 bytes free

C:\DATA>_

If you want to list all the files beginning with the letter *R* and whose extensions are TXT, use the filespec *r*.txt*:

C:\DATA>dir r*.txt

Volume in drive C has no label
Directory of C:\DATA

REPORT91	TXT	8485	1-18-91	11:20a
REVENUE	TXT	7357	3-18-91	12:00p

2 File(s) 10629120 bytes free

C:\DATA>_

Fortunately, if you want to use the DIR command to list the files beginning with a particular letter, you don't need to include the filename extension in your filespec. For instance, to list the files whose names begin with the letter *R*, simply use the filespec *r**:

C:\DATA>dir r*

and DOS will respond

Volume in drive C has no label
Directory of C:\DATA

REPORT	JAN	6434	2-13-91	10:04a
REPORT	FEB	6544	3-18-91	12:40p
REPORT	MAR	6134	4-11-91	3:30p
REPORT91	TXT	8485	1-18-91	11:20a
REVENUE	TXT	7357	3-18-91	12:00p

5 File(s) 10629120 bytes free

C:\DATA>_

Keep in mind, however, that all other DOS commands require you to supply a filename extension in your filespec. For example, when you want to use the COPY command to copy to drive A all the files whose names begin with the letter *R*, you can't use the filespec *r**. Instead, you need to use the filespec *r*.**, like this:

C:\DATA>copy r*.* a:

Specifying all files

As you've probably figured out, you can specify all the files in the current directory by using the filespec **.**. Of course, you don't need to use this filespec with the DIR command, since the command automatically lists all of the files in the current directory when you omit the filespec.

However, you often may need to supply the filespec **.** for other commands, such as COPY and DELETE.

A warning

Make certain you don't forget the following rule governing the asterisk wildcard: DOS will not recognize letters in the filespec that fall between the asterisk and the period separating the file's root name from its extension. For example, suppose you want to delete the files BUDGET91.TXT and REPORT91.TXT. Since these are the only two files whose root names end with 91, you might be tempted to issue the command

C:\DATA>del *91.*

If you issue this command, however, then use the DIR command to list the remaining files, you'll find that, much to your dismay, DOS has erased every file in the directory! Why? Because the characters 91 fall between the asterisk and the period in the filespec. In other words, the two commands

C:\DATA>del *91.*

C:\DATA>del *.*

are identical.

Fortunately, DOS provides a way to erase both BUDGET91.TXT and REPORT91.TXT with a single command. To do this, you'll need to use DOS' other wildcard character—the question mark.

The question mark wildcard

Unlike the asterisk wildcard, which can represent a variable number of characters from one to eight, the question mark wildcard can represent only one character in the filespec. You'll run into a few instances in which you'll need to use the question mark instead of the asterisk.

Let's look at a couple of examples that demonstrate the question mark wildcard. When you issue the command *dir report.?a?*, DOS displays only the files whose root name is REPORT, and whose extension begins with any letter, follows with the letter A, and ends with any letter:

C:\DATA>dir report.?a?

and DOS responds

Volume in drive C has no label
Directory of C:\DATA

REPORT	JAN	6434	2-13-91	10:04a
REPORT	MAR	6134	4-11-91	3:30p

2 File(s) 10629120 bytes free

C:\DATA>_

Now, let's revisit the example that gave us trouble a moment ago. As you'll recall, you wanted to operate on

the files BUDGET91.TXT and REPORT91.TXT. To do this, you can use the filespec ??????91.TXT. This filespec tells DOS to operate on any file that begins with any six characters, follows with the characters 91, and ends with the extension TXT. For example, when you use this filespec with the DIR command

```
C:\DATA>dir ?????91.txt
```

DOS will list these two files:

```
Volume in drive C has no label
Directory of C:\DATA

BUDGET91 TXT      3919  12-17-90  12:23p
REPORT91 TXT      8485  1-18-91   11:20a
    2 File(s)  10629120 bytes free
```

```
C:\DATA>_
```

To delete these files, simply issue the command

```
C:\DATA>del ?????91.txt
```

Be careful!

Whenever you're using wildcards to delete, copy, or rename files, be very careful. It's always a good idea to first use the DIR command to make sure that your filespec is operating on the appropriate files.

Conclusion

DOS offers two wildcard characters (* and ?) that let you operate on several files at once. Although you'll typically use the asterisk wildcard, which can represent as many as eight characters, you'll occasionally run into a situation that calls for the question mark wildcard. Be very careful when you use wildcards with the COPY, DELETE, and RENAME commands—it's easy to specify accidentally the wrong group of files. ■

COMMAND TECHNIQUE

VERSIONS
4.01-5.0

Creating a safer DEL command

If you've ever accidentally deleted an important file, you were probably upset, but you learned how dangerous DOS' DEL command can be. Unless you specify *,* when you issue the DEL command, DOS deletes the file and returns you to the system prompt without any warning.

Because DEL is so dangerous, Microsoft included a command in DOS 5 called UNDELETE, which lets you recover your recently deleted files, and a command called MIRROR, which keeps track of the File Allocation Table (FAT) for each file you delete. (For more information on these two powerful commands, see the articles "A Second Chance: How DOS 5 'Undeletes' Files" and "Protecting Your Disk with DOS 5's MIRROR Program" in last month's issue of *Inside DOS*.) However, UNDELETE and MIRROR won't always be able to recover deleted files.

Because removing the DEL command to prevent you from accidentally deleting files would be extreme, you need a better way. To this end, the DEL command in DOS versions 4 and 5 includes a switch, /P, that prompts you for confirmation before deleting the file.

The DEL command's /P parameter

For example, suppose you want to delete the file JODY.DOC from the \SMRTALEK directory. If you issue the command

```
C:\SMRTALEK>del jody.doc
```

```
C:\SMRTALEK>_
```

DOS deletes the file and immediately returns you to the prompt. However, if you include the /P switch after the filename, DOS asks you to verify the deletion:

```
C:\SMRTALEK>del jody.doc /p
```

```
C:\SMRTALEK\JODY.DOC, Delete (Y/N)?_
```

If you've typed the wrong filename or otherwise want to keep the file, you now have a second chance. As a safeguard, you can train yourself to type the /P switch whenever you delete files. However, if you want to ensure DOS *always* prompts you (or anyone else), you can record a DEL macro.

The key to this technique is that when you issue a command, DOS looks for a macro by that name before it looks for the internal DOS command. This means that if you define a DEL macro, DOS will carry out this macro's instructions instead of the internal DEL command whenever you type *del*. Therefore, all you need to do to create a safer DEL command is to define a DEL macro that includes the /P switch.

Defining the DEL macro

To define the DEL macro, issue the following command

```
C:\>doskey del=del $1 /p
```

and press [Enter].

Let's take a look at what this command does. The

```
doskey del=
```

portion of the command tells DOS that you want to load the Doskey program and create a macro called DEL. The remainder of the command constitutes the commands DOS will carry out whenever you run the DEL macro: The *del* statement to the right of the equal sign is the internal DOS DEL command; the characters \$1 constitute a replaceable parameter, enabling you to substitute any filename when you run the macro; and the /P switch tells DOS to verify the deletion.

Running the macro

Once you create the DEL macro, DOS will always prompt you for verification whenever you delete a file. In other words, you (or anyone else) will always get a second chance when deleting files. For example, instead of simply deleting the file and returning you to the system prompt, the command

```
C:\SMRTALEK>del jody.doc
```

now prompts you for verification:

```
C:\SMRTALEK\JODY.DOC, Delete (Y/N)?_
```

Storing the macro

As you may know, DOS stores Doskey macros in RAM, which means that unless you save your macros, you lose them when you turn off your computer. To load the DEL macro each time you boot up, you can place the statement

```
doskey del=del $1 /p
```

in your AUTOEXEC.BAT file. Another option is to store the macro along with other Doskey macros you create in a MACROS.BAT batch file, then call MACROS.BAT from

AUTOEXEC.BAT by including in AUTOEXEC.BAT the line
call macros

This technique has two advantages. Because you store all your macros in a separate batch file, your AUTOEXEC.BAT file doesn't become cluttered with macros. Second, you don't need to edit AUTOEXEC.BAT each time you add a macro. For a detailed discussion of storing and calling your macros each time you turn on your computer, see the article "Storing Your Macros and Automatically Loading Them at Bootup" in the September 1991 issue of *Inside DOS*.

Notes

Even if you like living on the edge when it comes to deleting files, you can still keep the DEL macro around for a couple of reasons. First, because the internal DEL command is the same as the internal ERASE command, you can use ERASE anytime you want to avoid the warning prompt. Second, if you're pretty sure of yourself, but you share a computer with someone else, you can use ERASE and he or she can use DEL.

Finally, if you want to delete all the files in a directory and avoid the warning DOS automatically provides with the *del ** command, see the article "Getting Around the DEL Command's Verification Prompt" in the December 1991 issue of *Inside DOS*.

Conclusion

DOS versions 4 and 5 provide a switch for the DEL command that prompts you for verification before it deletes files. This switch can make the DEL command less hazardous. In this article, we showed you how to take advantage of this feature and how to create a Doskey macro in DOS 5 that makes verification a permanent part of the DEL command. ■

VAN WOLVERTON

VERSIONS
2.1-5.0

Treat your hard disk with respect—and care

When hard disks became widely available at affordable prices, personal computer users felt a sense of freedom they hadn't experienced since video displays and keyboards replaced teletype terminals and paper tape. Not only did hard disks free us from the drudgery of swapping disks, they made possible the development and distribution of more sophisticated programs that processed larger quantities of data. How would *you* handle a 6-megabyte graphics file with 360K floppies?

But engineers giveth and engineers taketh away: Hard disks have made our computer work at once much easier and much riskier. Now all our eggs are in one basket—

when we stored data on floppies, it would have taken fire or theft to destroy it all; now all it takes is one hard disk failure.

We can choose from a host of available utility programs to protect our data, but we don't need them to make efficient use of our hard disks and protect ourselves against disaster. DOS provides help in three areas: organizing our file system; backing up our disk files; and monitoring our hard disk performance with CHKDSK.

Organize those files

You don't throw all your paper files in a big box—you organize and categorize, file and label them so that six

months from now you can find that insurance policy or canceled check. (You *do* keep up with your filing, don't you?) You shouldn't put all your disk files in one or two directories, either, for the same reason: As your collection of files grows, it gets harder and harder to find the ones you want unless they're stored in some orderly fashion.

That's why DOS includes a multi-level filing system, so that you can organize your disk storage to match the way you use your computer. Since version 4, DOS has included a Setup program that installs DOS in a directory named \DOS. Virtually every major application program you buy today installs itself in its own directory. Some even create subdirectories where you store your data files.

Version 2.0 of DOS introduced the multi-level filing system and was the first to support a hard disk. Root directories filled with hundreds of files were commonplace on hard disks until people figured out what directories were for and how to manage them.

As a rule of thumb, a directory where you store data files shouldn't contain more than 30 or 40 files. If you're having trouble finding files in some of your directories, it's time either to archive some files and delete them from your hard disk, or to define some additional categories of data files and create directories for them. If you're just starting to use your hard disk, lay out a directory structure now, before you start storing files willy-nilly; the time you spend now planning how to store your files will pay off every time you use your system.

Backing up will save you grief

If organizing the directories on your hard disk is the first thing you should do, backing up is the thing you should do regularly. The remarkable convenience and power of a hard disk is matched only by the risk it represents: If you store all your files there, a single hardware failure can wipe them all out.

Programs like Fastback Plus and Norton Backup do the job quickly and intelligently; if you have them, make sure you use them. But it isn't necessary to invest the money and learning time these programs require in order to protect your data; you can give yourself adequate protection using nothing more than the DOS BACKUP or XCOPY command.

Backing up with DOS doesn't offer the sophisticated features of backup programs, but the price is right. And don't believe everything you might have heard or read about a DOS-based backup procedure being slow or inefficient. Unless you'll be backing up 15 megabytes or more of files every day, the backup procedure will take the same amount of time regardless of what program you use—including DOS.

It won't take you more than an hour or two to set up a batch file that backs up all the changed files from the directories where you store data files. The backup process itself shouldn't take more than four or five minutes a day, tops. It's the cheapest insurance you'll ever buy.

The blessing can also be a curse

Hard disks are remarkably tough and reliable, especially considering the close tolerances and high-speed moving parts involved. But these very qualities often lull us into a false sense of security, because months or years may pass without any hard disk problems. But the laws that govern hard disk behavior were written by a fellow named Murphy, so of course the dreaded disk failure always happens right after we quit backing up.

Even though the hard disk is tucked away inside the case of the machine, we can still perform some preventive maintenance to reduce losing both performance and data. Again, as with backing up files, available utility programs expand on what DOS offers. These programs, in fact, offer capabilities that have no DOS counterpart.

Check your disk periodically

Programs such as Spinrite, Speed Stor, Norton Utilities, and Mace Utilities let you check every detail of your disk's operation, such as rotational speed and seek time; they also examine minutely the recording surfaces of the disk to see how reliably they can store data. Many of these programs are sensitive enough to detect a problem before it gets severe enough to cause a loss of data, giving you time to repair or replace the disk before, instead of after, a disaster occurs.

DOS itself provides only one command that lets you check your hard disk's condition: CHKDSK. It doesn't have the capabilities of the third-party programs, but it offers help with two problems:

- If a file is stored in widely separated areas of the disk, it takes DOS longer to read from and write to the file.
- Program malfunctions or power-line problems can create indexing errors in the storage locations that DOS maintains on the disk, making the corresponding parts of the disk unavailable to DOS.

CHKDSK can identify both of these problems, and help you solve one of them.

The case of the fragmented file

The hard disk is divided into small areas, called sectors, of 512 bytes each. Depending on the size of your hard disk, DOS assigns space for a file in chunks made up of one, two, or more sectors; these chunks are called *allocation units*.

When you store a file on a disk, DOS tries to store the entire file in the first block of available space. This works fine when the disk is new, but when you change a file, DOS erases the old version and stores the new one. As you create new files and change existing ones, the available space starts getting cut up into smaller and smaller pieces.

At some point, you'll save a file that's bigger than the first available block of space, so DOS saves as much as it can there and then looks for the next available block. Depending on how large the file is and how chopped up the available disk space is, the file could be stored in two, three, or more pieces. Such a file is said to be *fragmented*. Just as it takes more time to buy six items at six different stores than it would take to buy them all at one store, it takes DOS much more time to read or write a fragmented file.

Specifying a filename with CHKDSK

The CHKDSK command checks the files you specify as a parameter to see if they're fragmented. The following command, for example, would check all the files in the directory C:\WORD\REPORTS for fragmented files:

```
C:\>chkdsk \word\reports\*.*
```

If there are no fragmented files, CHKDSK responds:

All specified file(s) are contiguous.

If it finds fragmented files, CHKDSK identifies them after the memory report:

```
Volume RUBICON ONE created 08-05-1991 7:25a
Volume Serial Number is 2337-1CC9
```

```
340582400 bytes total disk space
 90112 bytes in 3 hidden files
1343488 bytes in 164 directories
141066240 bytes in 3904 user files
 40960 bytes in bad sectors
198041600 bytes available on disk
 8192 bytes in each allocation unit
41575 total allocation units on disk
24175 available allocation units on disk
```

```
655360 total bytes memory
582960 bytes free
```

```
C:\WORD\REPORTS\BUDGET.DOC Contains 3 non-contiguous blocks
C:\WORD\REPORTS\TRAVEL.DOC Contains 2 non-contiguous blocks
```

You need to repeat the command for each directory you want to check for fragmented files.

Restoring lost vigor

In many cases, you won't notice any slowdown caused by fragmented files; if you're satisfied with your system's performance, forget about fragmentation. But if your system has seemed more sluggish and the CHKDSK command tells you that you have a lot of fragmented files, you can regain that lost performance by storing all the files in adjoining (or contiguous) allocation units.

The DOS solution is a bit painful: You must back up your entire hard disk, reformat it, then restore all the files. The disk-optimizer programs offer a more convenient—albeit more expensive—solution. They will compact (or defragment; take your choice) all the files on your hard disk, restoring its previous unfettered speed. Depending on the size of your hard disk, this can take up to half an hour, but you don't have to sit there and watch.

Sectors lost, sectors gained

The other disk problem CHKDSK can identify doesn't affect the speed of the disk, but rather its capacity. DOS stores an index to each allocation unit on the disk. It uses this index, called the *File Allocation Table*, to keep track of where files (or, if they're fragmented, the parts of files) are stored.

If a program is written incorrectly, it can cause DOS to store incorrect data in this index. The index data can also be damaged if there is a power-line problem—a brownout or a complete failure—while the drive is writing data in the index. If this happens, DOS may lose track of where all or part of a file is stored, or may believe that a portion of the disk is being used when, in fact, it isn't.

CHKDSK always identifies such a problem, and can solve it by correcting the data in the index and gathering all the disk space thus made available into one or more files in the root directory. Because the solution requires changing the data in a sensitive part of the disk, however, CHKDSK requires that you explicitly request the change by specifying the /F parameter, then prompts for confirmation before actually doing it.

The puzzling /F parameter

To carry out the CHKDSK command and ask that any lost space be restored, you type:

```
C:\>chkdsk /f
```

If CHKDSK finds lost space, it tells you how many problem areas it found and how much space it could restore:

```
128 lost allocation units found in 2 chains.
Convert lost chains to files (Y/N)?
```

If you wanted to restore this space to use, you would press Y. DOS would create two files in the root directory named FILE0000.CHK and FILE0001.CHK. You could look at the files with an editor or word processor and save any valuable data, then delete the files and the space they occupied would become available.

Probably the most confusing aspect is that CHKDSK displays the messages identifying the lost space and asking if you want to restore it, whether or not you specify the /F parameter. If you didn't specify /F, CHKDSK doesn't

restore the space *even if you do press Y in response to the prompt.*

The diagnostic capabilities of DOS are so limited that several companies are doing a very nice business selling software that gives you the ability to monitor and correct several different types of problems that might afflict your hard disk. CHKDSK is minimal protection; if the data on

your hard disk is valuable, you should buy one of the programs that does the job right.

But there's no excuse for not organizing your data correctly or not backing it up—no program can do that for you. Although short on bells and whistles, DOS offers all the capability you really need to protect yourself against a disk failure. ■

PRODUCT REVIEW

VERSIONS
3.3-5.0

Double your hard disk space with Stacker 2.0

By Tim Landgrave, Consulting Editor

Remember when you first loaded up your new computer's hard disk with all those nifty programs? After you stored all those installation disks, you thought you'd never fill up that huge 30Mb hard disk. You had 20 million bytes left. You couldn't possibly write, analyze, or collect that much data.

Now your directory listing shows 20 bytes left. How did it happen? Well, first of all, programs keep getting bigger. Also, many machines not only have 1.5 megabytes of DOS to contend with, but also 3 megabytes worth of Windows. In this article, we'll look at some strategies for reclaiming lost disk space and then show you how to solve the problem with a third-party utility called Stacker.

Getting back some disk space

As your disk begins to fill up, you can employ several different tactics for managing disk space. First, you can erase applications you no longer use and reinstall them when you need them. However, you may use some applications frequently for short amounts of time. For example, if you only use a spreadsheet at the end of the month to update your budget, you can erase the program and reinstall it when you need it. But you'll probably find it more convenient to back up the application than to erase and then reinstall it. Many times, installing a program is only half the battle. You still have to configure it for your display and printer. Using the DOS BACKUP command to remove the application and the RESTORE command to re-install it saves you the time involved in reconfiguring the application.

You can also save hard disk space by storing your data files on floppy disks. You can configure most applications to save files to the hard disk or on a floppy disk. Storing the file on a floppy disk gives you an additional advantage—your data will be safe if your hard disk crashes. But if your data files are too large to fit on floppy disks and it takes too long to back up and restore infrequently used applications, you should consider using a file compression program.

Using data compression

You can find several inexpensive data compression packages available in the software market. A data compression package analyzes your files and uses a variety of different methods to save the file using less disk space. These methods are 100% reliable and, depending on the type of file, allow you to store a file in less than half the disk space. These programs can compress text, worksheet, and database files down to one-third to one-half their original size. But even after compression, executable files consume three-fourths of the disk space they required before compression.

Also, you need to decompress any programs or data files before you can use them. To do this, you need enough space on the hard disk to store them in their decompressed form. Since you'd like the advantages of compression but the convenience of using your files without manually decompressing them, the ideal solution is to let DOS dynamically compress and decompress programs and data files when it needs them. That's what Stacker does.

What is Stacker?

Stacker is a device driver from STAC Electronics which works with DOS to double the amount of disk space available to your programs. It works with all versions of DOS starting with 3.3 and Microsoft Windows 3.0. After installing Stacker, your 30, 40, or 80 megabyte hard disk can store 60, 80, or 160 megabytes of data, respectively. Stacker performs this magic by intercepting a program's DOS request for disk space and filling the request using its own device driver. Although Stacker normally uses your computer's microprocessor to perform the compression, you can also purchase a Stacker coprocessor card to perform the compression work. Let's look at what Stacker does in more detail.

How Stacker works

When you install Stacker, you have the option of *stacking* an empty disk or a partially filled disk. Stacking is the

process of compressing the data on your disk into a single file.

First, Stacker compresses your data into a single hidden file called

STACVOL.XXX

where XXX is DSK for the first partition and a number between 000 and 999 for additional partitions. Next, it installs the Stacker device driver in your CONFIG.SYS file. Finally, you reboot your machine for the changes to take effect.

After your PC restarts, you'll notice that you have at least one additional hard drive. That is, if you had a drive C before, you now have a drive D and maybe a drive E as well. When using your applications or data, you may need to refer to drive D or E instead of drive C. This is the only change you'll notice when using Stacker, except that you now have twice the disk space available.

Using Stacker with versions of DOS prior to 4.0 will create some problems if you have a disk drive with multiple partitions. In this case, you will have several hard drive letters to contend with and Stacker spreads your program and data files across them. Although the benefits of Stacker outweigh this disadvantage, you might want to consider upgrading to DOS 5 before installing Stacker.

What does it cost?

You should weigh several factors when deciding whether to purchase Stacker. First, while you won't take much of a performance hit for using Stacker on a 386 machine, you

should consider purchasing the coprocessor version for your 8086 or 286 machine. Most 386 machines can handle the additional processing overhead that Stacker imposes, while older processors can use the coprocessor's help. Also, Stacker uses conventional memory to store the TSR. If you have EMS memory (or an EMS emulator like EMM386), Stacker can use it to reduce the required amount of conventional memory. Stacker uses 26Kb of RAM if you have EMS memory available and 41Kb without EMS memory for the software-only version. The coprocessor version requires 14Kb of RAM if you have EMS memory and 30Kb without it. Finally, although Stacker will work with versions of DOS prior to 5.0, if you have a 386 machine, you should consider upgrading to DOS 5 or using Stacker with products like Quarterdeck's QEMM. This will allow you to load the Stacker device driver into high memory and emulate EMS memory for Stacker's buffers.

With a street price of under \$100 for the software-only version and \$200 for the coprocessor version, Stacker is an inexpensive option for increasing the amount of available hard disk space. You could buy a new hard disk for anywhere from \$300-\$500, but even after buying a new disk it makes sense to double its capacity by using Stacker. When considering how much hard disk space you should have for your system, remember Murphy's PC disk space law: "Computer programs and data expand to fill the disk space available." ■

For more information on Stacker, contact STAC Electronics at (619) 431-7474.

Continued from page 1

Loading an application and a file...

as shown in Figure A. If your DOC files are located in a directory other than the one containing the Word program file, you must type the full pathname of the program's executable file (in Word's case, \WORD5\WORD.EXE). Once you've made the association, you can load Word and any file bearing the DOC extension in a single step—we'll show you how in a moment.

Associating the extension with the program

If you prefer, you can select a program's executable file, then tell DOS which filename extension to associate it with. The technique is the reverse of the one we described in the previous section: You select the program's executable file, then type the extension in the Associate File dialog box. This technique comes in handy when you want to associate more than one extension with a program. For example, Microsoft Works, an integrated software pack-

age, uses different default filename extensions for its various environments. Using this second method, you can assign more than one extension to the program by typing the extensions in the Associate File dialog box separated by spaces. Let's illustrate this method by associating the extensions WDB, WKS, and WPS with Microsoft Works.

Begin by selecting the \WORKS directory in the Directory Tree window. When DOS displays the \WORKS directory's files in the File List window, locate and select the WORKS.EXE file. Next, choose the Associate... command on the File menu. This time DOS will present a slightly different Associate File dialog box from the one shown in Figure A. In the Extensions text box, type

wdb wks wps

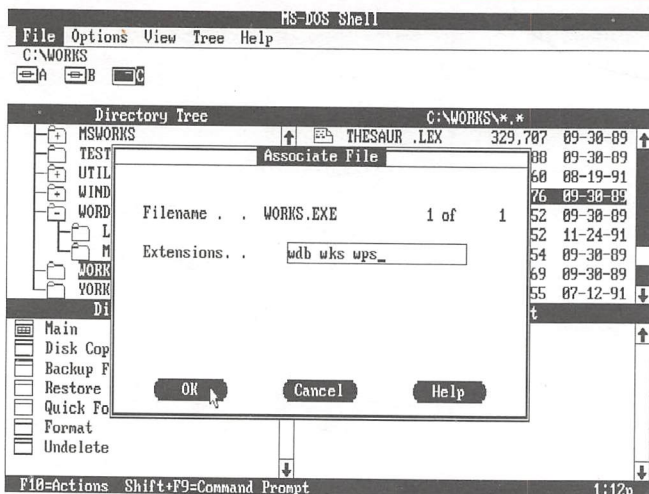
as shown in Figure B on the last page, and press [Enter] or click OK.

MS-DOS 5 Technical Support

(206) 646-5104

Please include account number from label with any correspondence.

Figure B

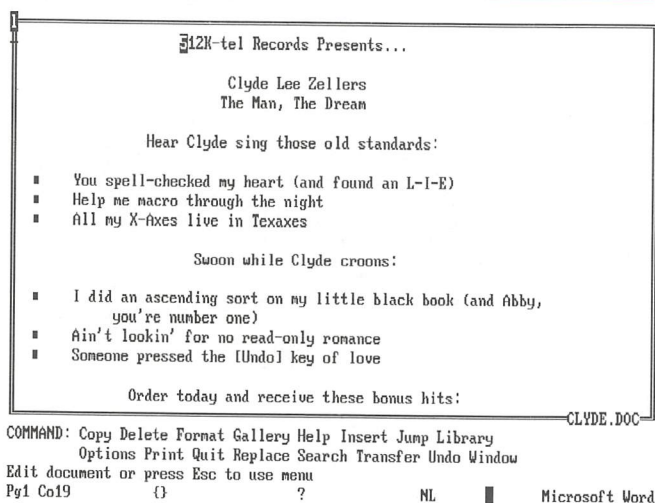


You can associate multiple filename extensions with a single application.

Launching the program and opening the file

Now that you've associated a filename extension with a program, launching the program and opening a file at the same time is a breeze. All you need to do is double-click on any filename in the File List window containing the appropriate extension, or select the filename with the keyboard,

Figure C



Once you associate an extension and an application, opening the file is as easy as double-clicking on the filename.

and press [Enter]. When you do, DOS will load the application and open the file whose name you selected. For example, because we associated the DOC extension with Microsoft Word, DOS automatically loaded Word and opened the file CLYDE.DOC when we double clicked on its filename in the File List window, as shown in Figure C.

Notes

There are a couple of things to keep in mind when you associate a program with a filename extension. First, you can only associate a filename extension with one program at a time. In other words, once you've associated the DOC extension with Microsoft Word, you can't associate it with another program without losing the initial Word association.

Second, you might want to "unassociate" an extension. To do so, simply select any file containing that extension, and issue the Associate... command. When DOS presents the Associate File dialog box, press [Backspace] and then press [Enter].

Conclusion

DOS 5's Shell allows you to "associate" a filename extension with an application program. Once you make such an association, you can easily launch the application and open any file bearing that extension in one simple step. If you work from the Shell, you'll want to take advantage of its Associate... command.

Version icons

With the advent of MS-DOS version 5, there are several DOS versions being used today. Because these versions of MS-DOS and PC-DOS feature many different commands and operations, some techniques we present won't apply to all versions.

To make it easier to ascertain which articles apply to your version of DOS, we'll place an icon at the top of each article identifying the version (or versions) the article discusses.

Within the articles themselves, we'll make it clear as to which version we're referring whenever the distinction is critical.

